10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 30 August – 1 September 2004, Albany, New York

Reconfigurability in MDO Problem Synthesis, Part 2

Natalia M. Alexandrov[*]

NASA Langley Research Center, Hampton, Virginia, 23681-2199, USA

Robert Michael Lewis[†]

College of William & Mary, Williamsburg, Virginia, 23187-8795, USA

Integrating autonomous disciplines into a problem amenable to solution presents a major challenge in realistic multidisciplinary design optimization (MDO). We propose a linguistic approach to MDO problem description, formulation, and solution we call reconfigurable multidisciplinary synthesis (REMS). With assistance from computer science techniques, REMS comprises an abstract language and a collection of processes that provide a means for dynamic reasoning about MDO problems in a range of contexts. The approach may be summarized as follows. Description of disciplinary data according to the rules of a grammar, followed by lexical analysis and compilation, yields basic computational components that can be assembled into various MDO problem formulations and solution algorithms, including hybrid strategies, with relative ease. The ability to re-use the computational components is due to the special structure of the MDO problem. The range of contexts for reasoning about MDO spans tasks from error checking and derivative computation to formulation and reformulation of optimization problem statements. In highly structured contexts, reconfigurability can mean a straightforward transformation among problem formulations with a single operation. We hope that REMS will enable experimentation with a variety of problem formulations in research environments, assist in the assembly of MDO test problems, and serve as a pre-processor in computational frameworks in production environments. Part 1 of two companion papers, discusses the fundamentals of REMS. This paper, Part 2 illustrates the methodology in more detail.

I. Introduction

This paper provides further details of reconfigurable multidisciplinary synthesis (REMS), the approach to multidisciplinary design optimization problem formulation introduced in the companion paper, *Reconfigurability in MDO Problem Synthesis, Part 1.* As in that paper, our focus is on that part of the design process that can be formulated as an optimization problem.

In our view, there is a clear need for such a logical framework that will enable researchers and practitioners of MDO to reason about various problem formulations in view of the problem structure and the optimization algorithms available. The ideas reported here grew out of our own desire for such a tool in our research, and our experience trying to reconfigure test problems that have appeared in the literature.

The possibility of conceptually reconfiguring computational components in an automatic fashion to arrive at different problem formulations and algorithms for their solution flows naturally from the fact that most

[*]Senior Research Scientist, Multidisciplinary Optimization Branch, Mail Stop 159. Senior Member AIAA.

[†]Assistant Professor, Department of Mathematics, P.O. Box 8795. Member AIAA. This research was supported by the National Aeronautics and Space Administration under NASA Grant NCC-1-02029 and by the National Science Foundation under Grant DMS-0215444.

MDO formulations share the basic computational components, comprising output/input couplings and the attendant sensitivity information. Moreover, MDO problem formulations may be viewed as being related (or distinguished) by the specific closure of constraints and the elimination of independent variables in a suitably general abstraction of the MDO integration problem. Although the task of integrating disciplinary analyses is inherently difficult, MDO problem synthesis and solution can be significantly eased with an appropriate implementation of the basic computational components and a general abstract framework in which to reason about the problem.

In this paper we examine such an abstract framework and discuss its relation to MDO problem formulations. These formulations may be viewed as deriving from the way that independent variables are eliminated from the problem. The elimination of independent variables makes some of the variables in the problem to be treated as functions of the remaining independent variables. Variable elimination is achieved either through the use of implicit relations defined by equality constraints or through disciplinary optimization problems.

It is possible to develop a formal grammar that describes the underlying structure of the MDO problem, beginning at the level of the constituent subsystems, though we do not discuss this point in detail here. From these individual pieces we can then generate various intermediate representations (IR) of the pieces of the MDO problem and derive an understanding of what changes in the problem as a consequence of choosing different MDO formulations. This understanding is to be achieved in a systematic and automatic way by applying transformations to the underlying IR.

The key, in our view, is to develop a specification of MDO problems that begins at the lowest disciplinary or subsystem level and proceed to the MDO problem level via the incremental introduction of additional information. This allows one to begin with appropriate descriptions of the subsystems, described in isolation, and reason upward in a systematic way to various MDO problem formulations.

The basic ideas and technology are like those used in computer language compilers and also in automatic differentiation (which is itself based on compiler technology), and we make no claims of novelty for the basic mathematical tools we apply. Moreover, there are similarities between elements of the proposed approach and work that has appeared previously (see the companion paper[1] for a brief review); however, our goals differ.

Our experience attempting to modify extant test problems for our own purposes is that the low-level structure we need to know in order to reformulate the problem typically has been obscured in the test problem we inherit. The multidisciplinary problem structure has been completely integrated into a particular MDO problem formulation, and this makes it difficult to “dis-integrate” the problem for the purposes of reformulation.

II. The data flow graph

We model the flow of data among the constituent disciplines and the optimization objectives and constraints as a directed graph, or digraph, a widely used abstraction in applied mathematics.[2,3] A directed graph consists of a set of nodes, a set of edges between nodes, and a two relations of incidence. For each edge, the incidence relations give the directedness of the edge: given an edge between two nodes, one node is distinguished as the head of the edge and one as the tail.

We denote graphs generically by G. Given a node $n \in G$, we speak of the *predecessors* and *successors* of n. The predecessors of n, which we denote by $P(n)$, are the nodes m for which there is an arc from m to n. The successors of n, which we denote by $S(n)$, are the nodes m for which there is an arc from n to m.

The fundamental types of nodes in the data flow graph are the *data nodes*, which we denote generically by d, and the *function nodes*, denoted generically by f. The data nodes represent the inputs and outputs of functions, e.g., the inputs and outputs of a disciplinary analysis. The function nodes represent the optimization objectives and constraints, and, most significantly, the various subsystem operations. A function node may represent an operation as large as an entire disciplinary analysis, but might also be a very simple, low-level operation. A hierarchical level of detail is possible, with a given function node representing a collection of functions and data nodes, with the desired level of detail made visible.

Data are represented by nodes rather than by arcs since a given datum may be the input to more than one function. Each data node has a single predecessor or successor, either a function or a data node. As we shall see, an edge between two data nodes is an opportunity for decoupling. The predecessors of each function node are always data nodes.

A node with no predecessors is called a *root* or a *source*, while a node with no successors is called a *leaf* or a *sink*. Local and system design variables will typically be sources, while terminal outputs, such as optimization objectives and constraints, will be sinks.

Associated with a digraph is its *incidence matrix*. The incidence matrix A of a digraph G is constructed as follows. Index the nodes of G by $1, \dots, N$. Then A is an $N \times N$ matrix whose (m, n) entry is 1 if there is an arc from m to n, and 0 otherwise. We use both the digraph and the incidence matrix as intermediate representations of coupled MDO systems.

A *directed path* in a digraph is a sequence of nodes $n_1, \dots, n_p$ such that n_{i+1} is a successor of n_i for each $i = 1, \dots, p-1$. A directed path may also be viewed as containing the directed edges between the nodes that define the path. A *cycle* is a directed path whose first and last nodes are the same. We use the notation $m \prec n$ if there is a directed path from node m to node n. This means there is a dependence of m on n.

It is possible that $m \prec m$, which means that m lies on a cycle. If m is a data node, then $m \prec m$ indicates that m is defined by an implicit relation involving m. That is, $m \prec m$ indicates that an iterative process must be performed to compute the quantity m. Depending on the ultimate problem formulation, this may mean that there is a sequentiality in the computations represented by the data flow graph. There exist algorithms for detecting all the cycles in a digraph,[4] so cycle and feedback analysis can be performed automatically using the data flow digraph. In this paper we consider only the static data flow graph and do not discuss the issue of cycle and feedback analysis. We note, however, that the DeMAID system,[5] for instance, performs a similar analysis and will provide a basis for our future investigation of feedback analysis.

When considering the dependences of data on other data, it is useful to think of the data nodes by themselves. For this purpose, we define a related graph G_{data}, derived from G by deleting the function nodes and connecting the predecessors of each function node to the successors. That is, let $f \in G$ be a function node; then, for each $d \in S(f)$, we set $P(d) = P(f)$ in G_{data}.

III. Model problem

Figure 1 depicts an artificial MDO problem. We use this problem to illustrate the assembly of the data flow graph and its manipulation at the level of the IR of the problem. This level of abstraction enables us to hide most of the implementation and its complexity from those specifying the problem.

In Figure 1 there are four disciplines, the boxes labeled A,B,C,D. Each discipline has a set of inputs and outputs, denoted by the round data nodes. The input–output relations for each discipline are given for each discipline in isolation. Specification of the input–output relations at this level, without reference to any other discipline, gives the maximum flexibility in later analyzing the couplings and opportunities for reformulating the MDO problem. The labeling used in Figure 1 is purely a place-holder; in practice the functions and data are labeled by descriptive character strings.

Note that in the model of a discipline illustrated in Figure 1, the outputs of the discipline are specified without regard to where they may go in the multidisciplinary setting. The approach we propose is based on using the specification of disciplines in isolation as much as possible to determine the multidisciplinary structure.

Divorcing the disciplinary specification from the multidisciplinary context also simplifies the notation and mathematics. If we were to use, say, notation such as "u_{AB}" and "u_{AC}" to represent the outputs from discipline A that are passed to B and C, respectively, then there is the complication that some of the quantities in u_{AB} and u_{AC} may be the same. Sticking to the lowest-level, autonomous specification of a discipline avoids this complication.

In Figure 2, the data flow for the completely coupled system is shown as it exists for a multidisciplinary analysis. The inputs of each discipline have been associated with the corresponding output. Each colored area contains a discipline and its outputs. The interdisciplinary output-input relations are represented by the red arcs that leave a colored area and enter an area of a different color.

In Figure 3, we have added optimization components to the data flow graph. The optimization components are the objective F and the constraints C, $c_{\{A,B,C,D\}}$. The optimization formulation given in Figure 3 is one in which the optimization is superimposed in a conventional way on a multidisciplinary analysis. The nodes (F, C, $c_{\{A,B,C,D\}}$) are function nodes that represent optimization objectives and constraints. From Figure 3 we see that optimization components are sinks. To simplify the diagram, we have chosen to specify the optimization function nodes as having no data output, though in reality they actually would. Sources, on the other hand, represent design variables that are manipulated by the optimization algorithm being used or possibly static parameters. We can perform error checking at this point to make sure all data nodes that appear as sources or sinks are intended as such, or whether an output/input pairing has been misspecified.

Figures 4–8 describe a test problem taken from Sobieski, Agte, and Sandusky.[6] The disciplines are range, power, drag polar, and weight. Figure 4 describes the totality of input/output relations for the coupled system. Taken individually, the input/output relations in Figures 5–8 are much simpler.

As our examples illustrate, the subsystem input/output relations can be described in a straightforward way. The key is not to insist on knowing at this stage the interdisciplinary origins of the inputs or the interdisciplinary destinations of the outputs. This means that the input/output relations can be described autonomously at the subsystem level, an important feature in the MDO setting.

These low-level input/output relations can then be assembled in a systematic (automatic) way to arrive at the interdisciplinary couplings. At this assembly level we can perform some error checking; for instance, we can check to make sure that if an input is expected, then there is a matching output for it.

The model here is that disciplines pull information in, rather than push information out. In the discipline-push model, we must know *a priori* which outputs are sent where. However, this information is not needed to describe a discipline in isolation. In the discipline-pull model, the pairing of outputs with inputs automatically captures the coupling.

The process we propose for assembling the IR of the MDO coupling is very much like the compilation/linking steps of a computer language. The first step (compilation) is applied independently to each member of a collection of disciplinary input/output descriptions (source files) to produce an IR of each discipline (object files). The second step (linking) takes the IR of each discipline and combines them into an IR of the coupled system. We can then perform various transformations of the problem using the IR and study different MDO problem formulations.

IV. Interdisciplinary coupling and variable elimination

The interdisciplinary output-input data pairings are the points of flexibility in MDO problem formulation. While the multidisciplinary coupling depicted in Figure 2 or Figure 3 may seem the natural one, the real picture is that depicted in Figure 9. In Figure 9 we have not identified each input with the corresponding output and eliminated one of the nodes from the graph. Each of the output/input pairings is an opportunity for modifying the problem formulation, so we do not automatically eliminate one of the paired output/input nodes. The output labels in the output/input pairings are distinguished by bars.

Data nodes are typically viewed as being replicated in the MDO problem to allow autonomy in the disciplinary calculations. We would say that the replicated nodes are inherent in the disciplinary descriptions (Figure 1), and we keep this replication explicit in the data flow digraph representation. This reflects the fact that the problem we are considering is one of problem integration, rather than problem decomposition.

At this point, we can begin to apply abstract problem transformations. In the following discussion, for any letter v, we use v to denote the vector of outputs v_i. For instance, $a = (a_1, a_2, a_3, a_4)$.

For each output/input data pairing there is an implicit equality consistency constraint, simply

$$\text{output} = \text{input}.$$

If we make all of these consistency constraints explicit in the system level optimization problem we arrive at the following formulation:

$$\begin{array}{lll} \underset{s,\alpha,\beta,\gamma,\delta,a,b,c}{\text{minimize}} & F(a_4, b_1, d_1) & \\ \text{subject to} & C(s, a_4) \geq 0 & \\ & c_A(\tilde{a}_1, \tilde{a}_4) \geq 0 & a = \tilde{a}(b_1, c_2) \\ & c_B(\tilde{b}_1, \tilde{b}_3) \geq 0 & b = \tilde{b}(a_3, c_2, d_1) \\ & c_C(\tilde{c}_1, \tilde{c}_3) \geq 0 & c = \tilde{c}(a_1, a_2, b_2) \\ & c_D(\tilde{d}_1) \geq 0 & d = \tilde{d}(b_3, c_1, c_2). \end{array} \tag{1}$$

If we close all of the consistency constraints, i.e., all the output-input data nodes are combined, we arrive at the following fully integrated problem, corresponding to Figure 3:

$$\begin{array}{ll} \underset{s,\alpha,\beta,\gamma,\delta}{\text{minimize}} & F(a_4(s,\alpha,\beta,\gamma,\delta), b_1(s,\alpha,\beta,\gamma,\delta), d_1(s,\alpha,\beta,\gamma,\delta)) \\ \text{subject to} & C(s, a_4(s,\alpha,\beta,\gamma,\delta)) \geq 0 \\ & c_A(a_1(s,\alpha,\beta,\gamma,\delta), a_4(s,\alpha,\beta,\gamma,\delta)) \geq 0 \\ & c_B(b_1(s,\alpha,\beta,\gamma,\delta), b_3(s,\alpha,\beta,\gamma,\delta)) \geq 0 \\ & c_C(c_1(s,\alpha,\beta,\gamma,\delta), c_3(s,\alpha,\beta,\gamma,\delta)) \geq 0 \\ & c_D(d_1(s,\alpha,\beta,\gamma,\delta)) \geq 0, \end{array} \tag{2}$$

where there is an underlying multidisciplinary analysis that can be described as the process of solving the system

$$\begin{array}{rcl} a & = & A(b_1, c_2) \\ b & = & B(a_3, c_2, d_1) \\ c & = & C(a_1, a_2, b_2) \\ d & = & D(b_3, c_1, c_2). \end{array}$$

Note that the data flow graph in Figure 3 is a transformation, in a systematic way, of the graph in Figure 9. The transformation involves coalescing data nodes with an edge between them. Moreover, for each such operation, the chain rule and implicit differentiation enable us to keep track of the effect of these output–input identifications on the sensitivities of the quantities in the data flow graph. Finally, the formalism of the data flow graph allows us to perform these transformations and analysis automatically, working on the graph representation.

The effect of closing constraints on sensitivities is straightforward to compute. Suppose $\phi(x, u)$ is a vector or scalar valued function, and let

$$\Phi(x) = \phi(x, u(x)).$$

We assume that given x, $u(x)$ is the solution of some system

$$S(x, u(x)) = 0. \tag{3}$$

Then $\Phi(x)$ reflects what happens when u is eliminated as an independent variable via (3). It is a standard result that follows from implicit differentiation[7,8] that the derivatives of ϕ and Φ are related as follows. Let W be the *injection operator*

$$W = W(x, v) = \begin{pmatrix} I \\ -S_u^{-1}(x, v) S_x(x, v) \end{pmatrix},$$

where a variable appearing as a subscript denotes the partial derivative with respect to that variable. The *reduction operator* is the transpose W^T.

Let $\lambda = \lambda(x, u)$ be given by

$$\lambda = -\left(S_u(x, u)\right)^{-T} \nabla_u \phi(x, u).$$

The Lagrangian $L(x, u; \lambda)$ is

$$L(x, u, \lambda) = \phi(x, u) + \lambda^T S(x, u).$$

Then

$$\nabla_x \Phi(x) = W^T(x, u(x)) \nabla_{(x,u)} \phi(x, u(x)). \tag{4}$$

The quantity on the right-hand side of (4) is known as the *reduced gradient.*[7] We also have

$$\nabla^2_{xx} \Phi(x) = W^T \left(\nabla^2_{(x,u)} \phi + \nabla^2_{(x,u)} S \cdot \lambda \right) W. \tag{5}$$

The quantity on the right-hand side of the preceding equation is the *reduced Hessian*[7] of the Lagrangian.

The relationships (4) and (5) show what happens to sensitivities when independent variables are eliminated via relations of the form (3). In our example, the relations

$$\begin{aligned} a - A(b_1, c_2) &= 0 \\ b - B(a_3, c_2, d_1) &= 0 \\ c - C(a_1, a_2, b_2) &= 0 \\ d - D(b_3, c_1, c_2) &= 0 \end{aligned}$$

are each relations of the form (3). Moreover, the reduction operator for any relation or set of relations of this form can be readily computed and applied in an symbolic manner. For instance, using $S(a, b_1, c_2) = a - A(b_1, c_2) = 0$ to define a as an implicit function of the other variables, we have

$$W^T = \begin{pmatrix} I \\ \dfrac{\partial A}{\partial(b_1, c_2)}. \end{pmatrix},$$

where $\dfrac{\partial A}{\partial(b_1, c_2)}$ should be taken to mean the Jacobian of A with respect to all the remaining variables, but since A only depends on b_1 and c_2, only these entries are nonzero. Note that the reduction operations we describe here contain essentially the same information as the generalized sensitivity equations.[9]

Symbolic manipulation using variable reduction allows us to see how to take the disciplinary input/output sensitivities and calculate system sensitivities for the coupled system. This provides guidance in implementing sensitivity calculations from the subsystem computational components. Again, implicit differentiation tells us that the superset sensitivities required can be derived from the input/output specifications for the individual disciplines, describing only the local dependences of each datum on the others. The global sensitivities can then be constructed automatically, essentially via the chain rule. In this regard, the manipulations we describe here resemble the techniques of the forward mode of automatic differentiation.[10]

Alternatively, nodes can be combined and treated as a single supernode if the nodes are viewed as being aggregated as a single computational unit. Sensitivities for the aggregated nodes can then be computed. This is useful if the bandwidth of the data being passed among function nodes varies considerably. By adding attributes that described the size of the outputs and inputs it is possible to look for function and data groupings that reduce the cost of sensitivity calculations.

For instance, consider the disciplines A and C in Figure 9. If the outputs $\tilde{a}_1, \tilde{a}_2$ and $\tilde{c}_1$ that couple A and C involve a large number of variables, while the outputs $\tilde{a}_3$ and $\tilde{c}_2, \tilde{c}_3$ coupling A and C to B and D involve a small number of variables, it may make sense to close the consistency constraints for the variables coupling A and C, removing these coupling variables and the cost of the attendant sensitivity calculations from the problem. Analysis of the digraph and incidence matrix can shed light on ways to aggregate functions to reduce sensitivity costs.

Formally, the aggregation of a set of nodes requires that we first identify all the function nodes in the set, and consider the subgraph defined by the function nodes and all their successors (these will be data nodes). All data nodes in the resulting subgraph that are sinks can then be treated as implicit functions of the remaining data nodes in the complement of the subgraph. The subgraph function nodes and any non-sink data nodes in the subgraph which never leave the set of operations being aggregated can then be treated as a single supernode, inheriting the edges into and out of the set of subgraph function nodes.

V. System and subsystem optimization problems

In the formalism we have described, optimization constraints and objectives can be added explicitly, as function nodes. We need not make any *a priori* distinction between disciplinary objectives and constraints since this information can be detected via analysis of the data flow digraph. Equally important, constraints and objectives can be added incrementally, and their description is independent of the discipline specifications. Again, definition of the constraint and objective inputs pulls in data from the the appropriate nodes. The data flow digraph also enables us to examine the sensitivity calculations involved in the optimization problem.

A more interesting and difficult question is the automatic introduction of disciplinary optimization problems as a technique for eliminating variables. In the preceding section we discussed variable elimination using the equality constraints implicit in the output/input relations between the disciplines. Eliminating variables through disciplinary optimization problems, on the other hand, seems inescapably to require the introduction of elements not implicit in the problem description, and to do so in a way that allows for a generic, template-like definition of the disciplinary optimization problems.

We illustrate this question using Collaborative Optimization (CO).[11] We apply a version of CO to the discipline D in Figure 9 as follows. From the system-level problem, Discipline D receives values of its inputs c_2, c_3, and b_3, a value for the system-level design variables $s_D = s$, and a target value d_1 for its output $\tilde{d}_1$. For discipline D we then solve the disciplinary problem

$$\begin{array}{ll} \underset{\delta}{\text{minimize}} & \frac{1}{2} \left\| \, \tilde{d}_1(\delta; s, b_3, c_2, c_3) - d_1 \, \right\|^2 \\ \text{subject to} & c_D(\tilde{d}_1(\delta; s, b_3, c_2, c_3) \geq 0. \end{array} \tag{6}$$

This is an optimization problem in the local design variables δ alone. (By a local design variable we mean an input to discipline that is an input to no other discipline.)

The optimal value of the objective in (6) is then returned to the system-level problem as an equality constraint. The system-level problem must drive the value of this constraint to zero. The output from solving (6) is a function only of the targets s, b_3, c_2, c_3, d_1 passed to D; the local design variables δ are eliminated from the system-level problem.

Incorporating approaches, such as CO, that introduce disciplinary optimization problems as a mechanism for eliminating variables makes for a bit of a dilemma in any abstraction of the structure of an MDO problem. One solution would be to describe the disciplinary optimization problem and its solution as a function node, explicitly state the specific inputs and outputs to this function node, and aggregate the eliminated local design variables inside this node.

It would be more attractive, however, if a disciplinary optimization problem could be described as a generic template and introduced automatically, without the need for explicit specification of the inputs and outputs for each discipline. The digraph structural abstraction we have described currently admits a partial solution in this direction.

The key is to observe that the variables that can be eliminated by a disciplinary optimization problem are local design variables. Local design variables can be determined automatically by examining the data flow digraph; they are source data nodes with a single arc out of them. (System-level design variables correspond to source data nodes with more than one out-going arc.) Abstractly, the introduction of a disciplinary optimization problem can be viewed as replacing a disciplinary analysis, i.e., function node, and the local design variables with a new function node that represents the disciplinary optimization problem.

The inputs and outputs of this new function correspond, at least in their data flow, to the non-local inputs and outputs of the original function node. In addition, the disciplinary optimization problem will typically have as an output node a sink corresponding to a system-level consistency constraint. In this way it is possible to give a generic specification of a disciplinary optimization problem as a means for eliminating local design variables that admits automatic application, in the same manner that we eliminated coupling variables via equality constraints previously.

VI. Concluding remarks

The idea of the approach to specifying and reasoning about MDO problem integration we have discussed is to draw the widest set of useful information from the least effort—descriptions of the input/output relations of the individual subsystems, in isolation. Moreover, the structure of MDO problems and the techniques used in their formulation allow, in the large, for specification in a formal grammar whose interpretation and manipulation can be automated. At this stage, we have outlined the formal specification of the grammar in a form appropriate for compiler construction.[12] The development of the language and techniques for analysis and manipulation is a focus of our current research.

References

[1]Alexandrov, N. M. and Lewis, R. M., "Reconfigurability in MDO Problem Synthesis, Part 1 and Part 2," August 2004, Papers AIAA-2004-4307 and AIAA-2004-4308.

[2]Cormen, T. H., Leiserson, C., and Rivest, R., *Introduction to Algorithms*, MIT Press, 1990.

[3]Harary, F., *Graph Theory*, Addison–Wesley, 1994.

[4]Johnson, D. B., "Finding all the elementary circuits of a directed graph," *SIAM Journal on Computing*, Vol. 4, 1975, pp. 77–84.

[5]Rogers, J. L., "DeMAID/GA An Enhanced Design Manager's Aid for Intelligent Decomposition," *Presented at the Sixth AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, Washington, 1996*, AIAA paper 96–4157.

[6]Sobieszczanski-Sobieski, J., Agte, J., and Sandusky, Jr., R., "Bi-Level Integrated System SYnthesis (BLISS)," Tech. Rep. NASA/TM-1998-208715, NASA LAngley Research Center, August 1998.

[7]Gill, P. E., Murray, W., and Wright, M. H., *Practical Optimization*, Academic Press, London, 1981.

[8]Lewis, R. M., "Practical Aspects of Variable Reduction Formulations and Reduced Basis Algorithms in Multidisciplinary Design Optimization," *Multidisciplinary Design Optimization: State-of-the-Art*, edited by N. Alexandrov and M. Y. Hussaini, SIAM, Philadelphia, 1997.

[9]Sobieszczanski-Sobieski, J., "Sensitivity of complex, internally coupled systems," *American Institute of Aeronautics and Astronautics (AIAA) Journal*, Vol. 28, No. 1, 1990, pp. 153–160.

[10]Griewank, A., *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Frontiers in Applied Mathematics, SIAM, Philadelphia, 2000.

[11]Braun, R., *Collaborative Optimization: An architecture for large-scale distributed design*, Ph.D. thesis, Stanford University, May 1996, Department of Aeronautics and Astronautics.

[12]Cooper, K. D. and Torczon, L., *Engineering a Compiler*, Morgan Kaufmann Publishers, 2004.

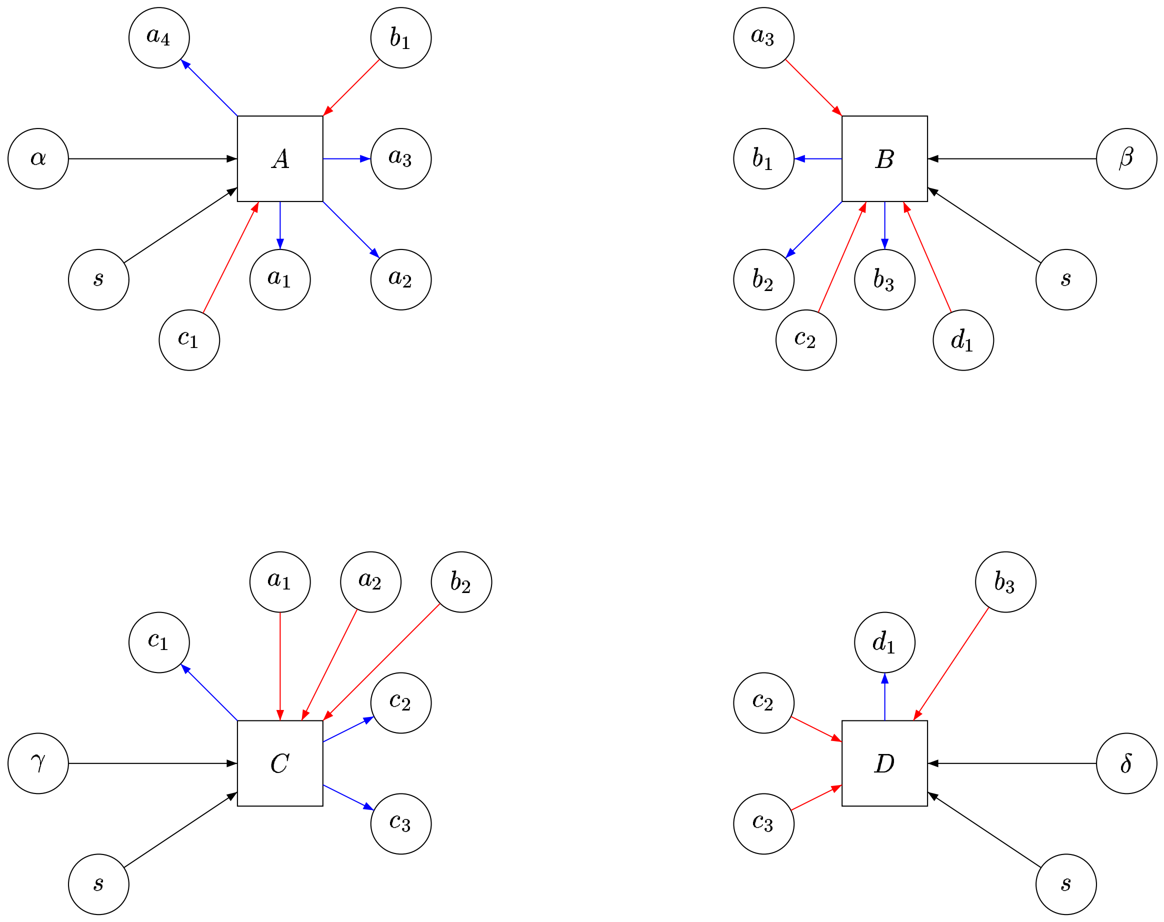

Figure 1. Input/output relations for the disciplines in isolation.

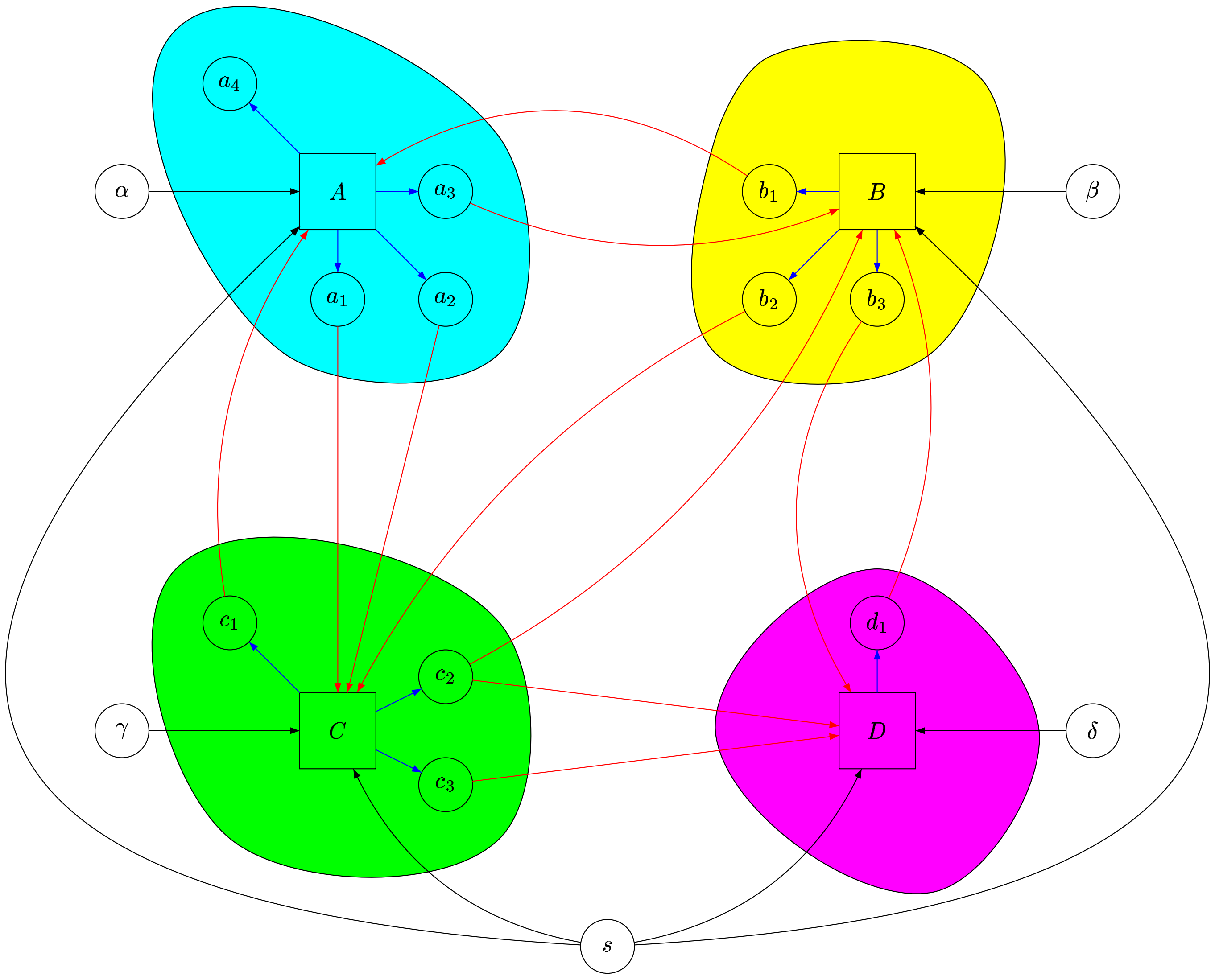

Figure 2. Input/output relations for the coupled system (no optimization).

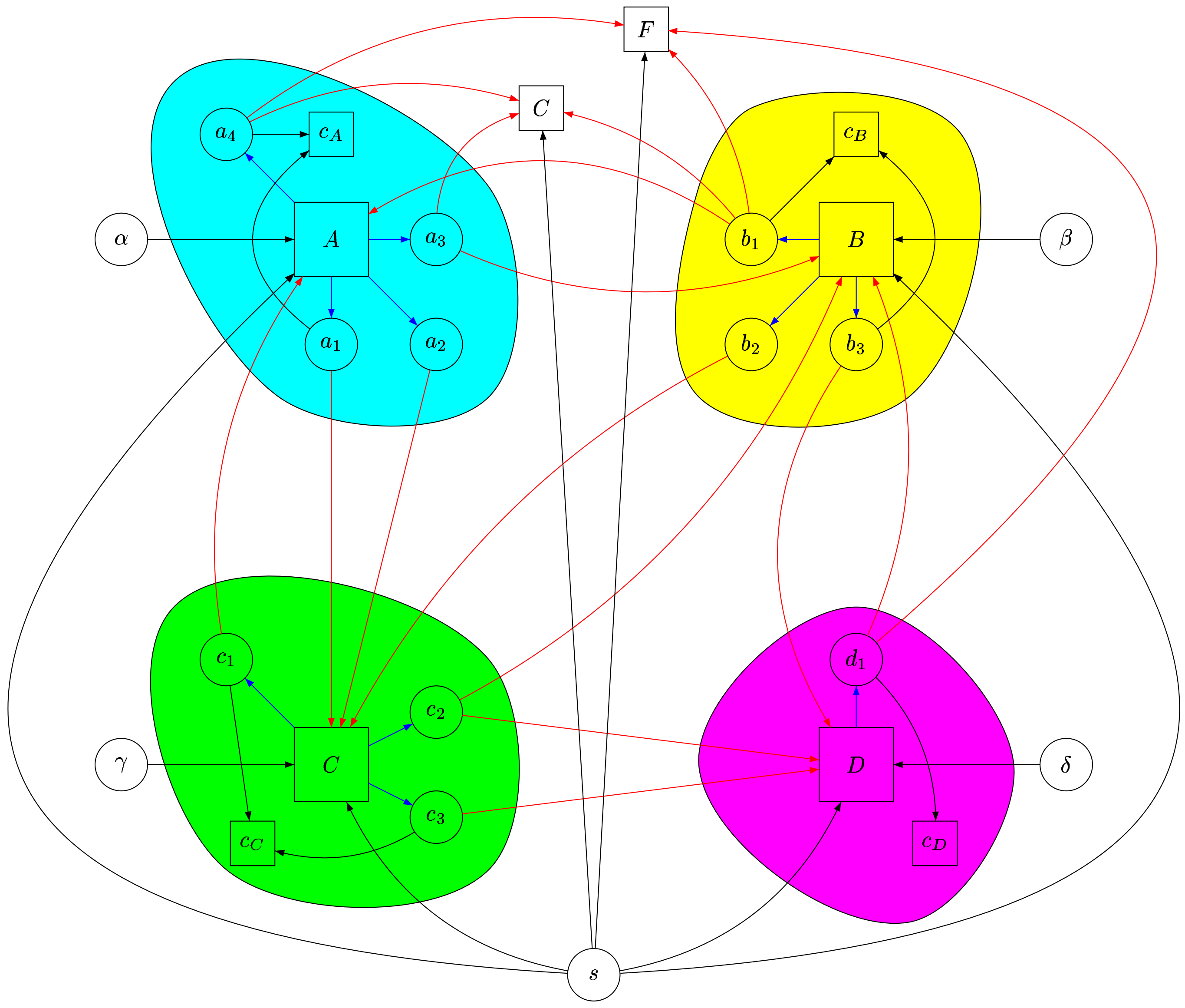

Figure 3. Input/output relations, fully integrated conventional NLP formulation based on a multidisciplinary analysis.

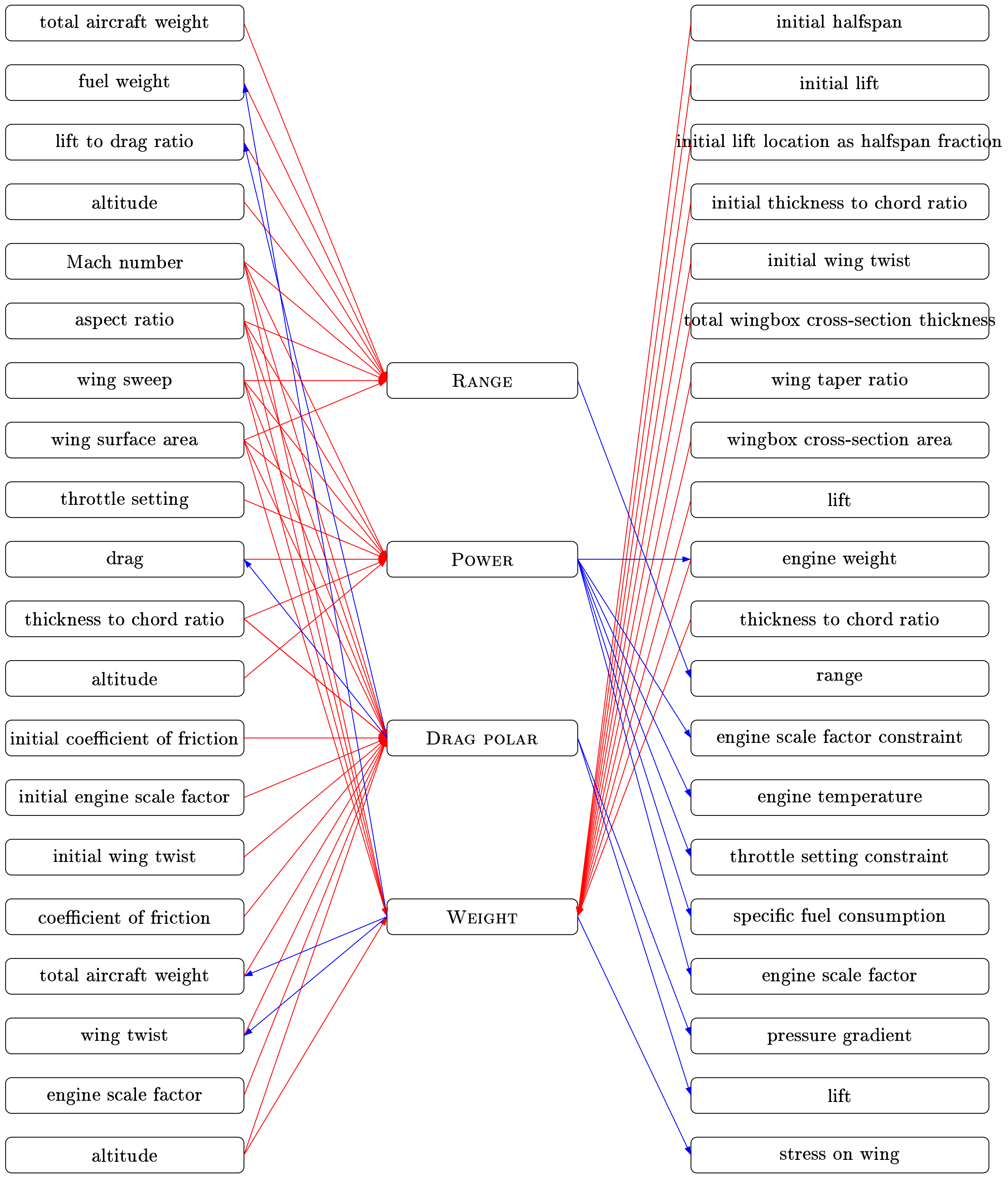

Figure 4. Comptutational graph of the coupled system

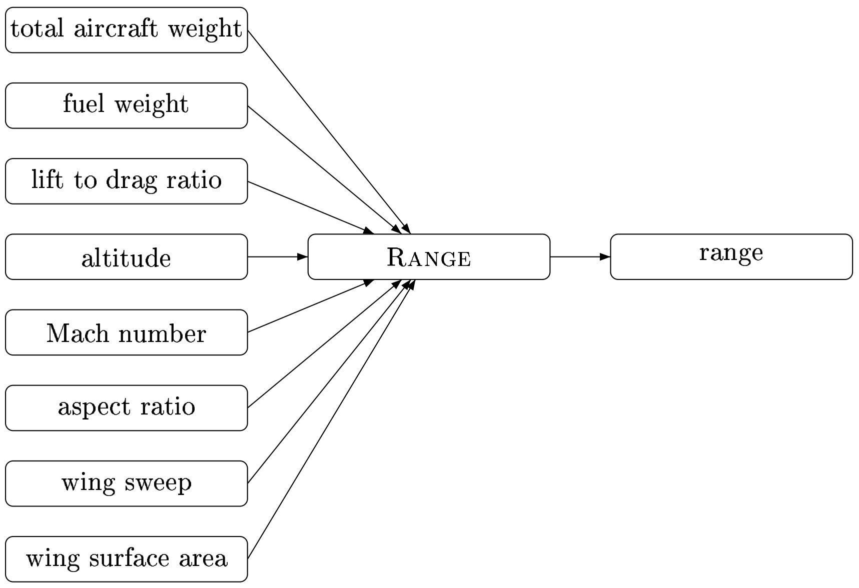

Figure 5. Input/output relations for the range calculations

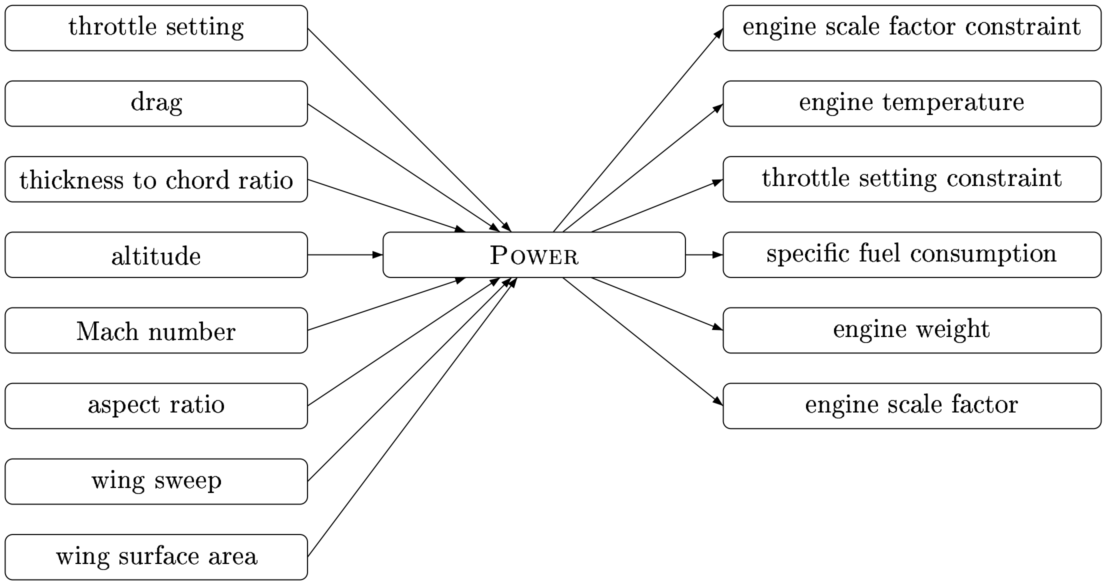

Figure 6. Input/output relations for the power calculations

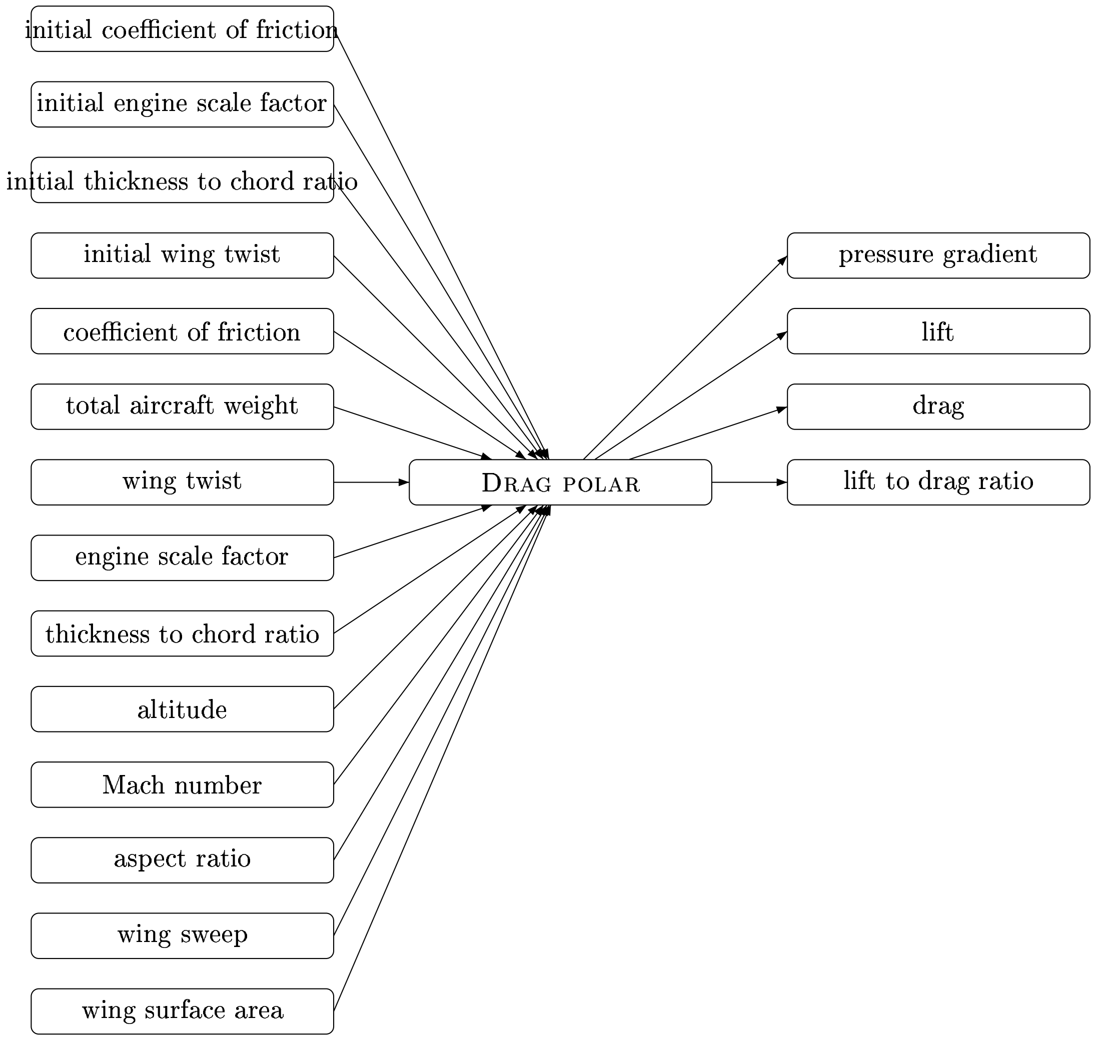

Figure 7. Input/output relations for the drag polar calculations

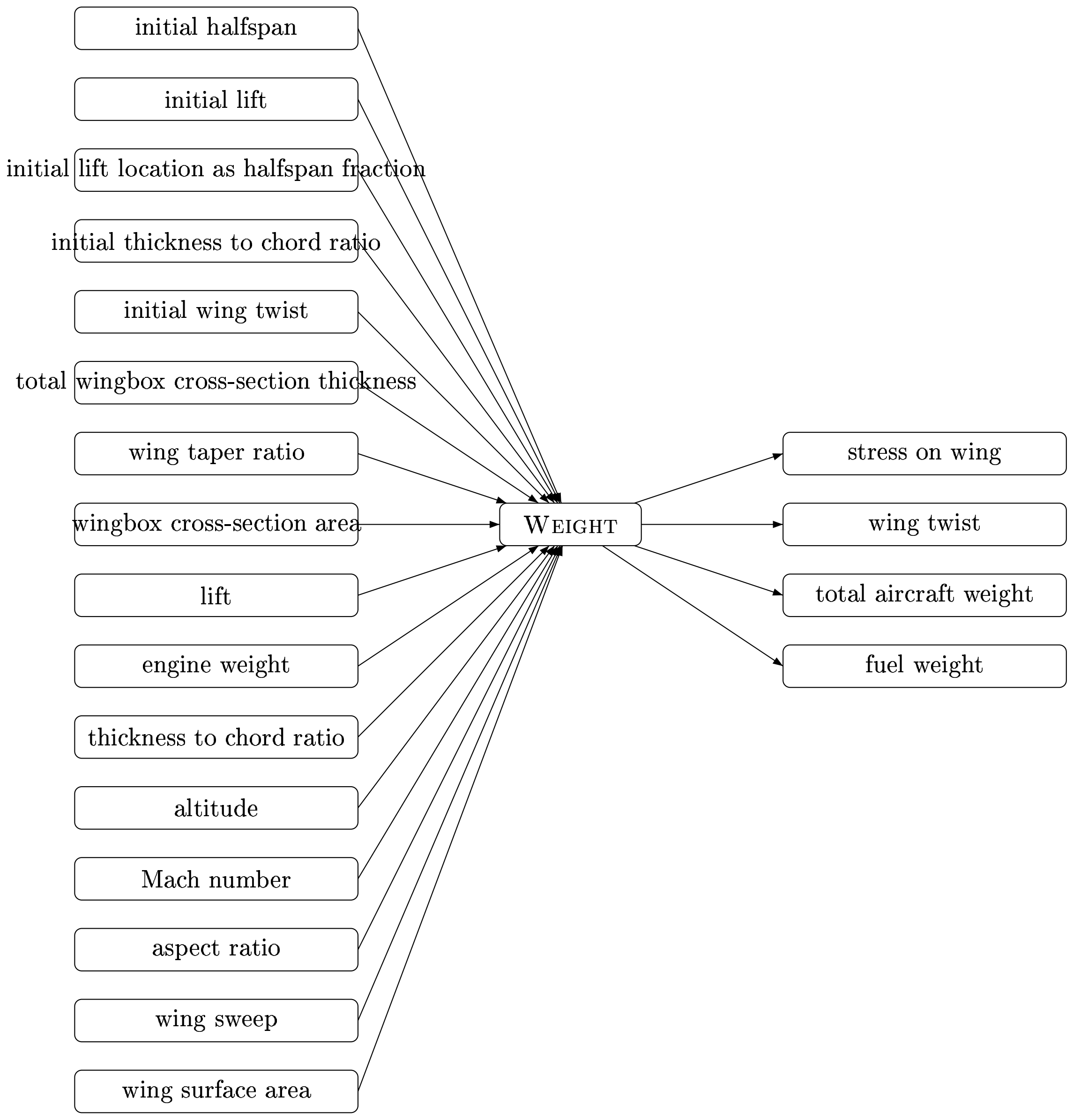

Figure 8. Input/output relations for the weight calculations

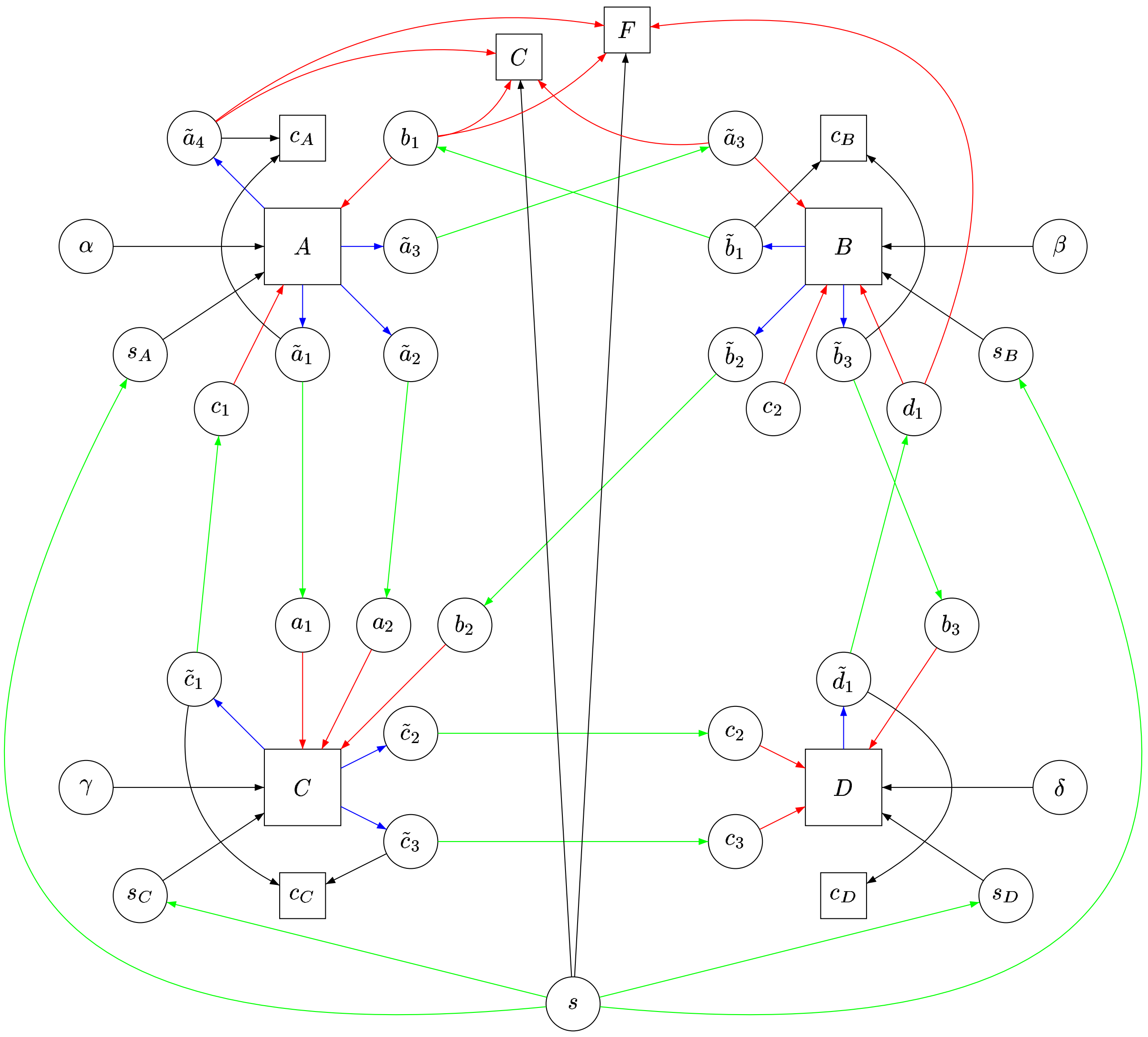

Figure 9. Data flow for the coupled disciplines.